BAB II
LANDASAN TEORI

Pada bab ini dilakukan pembahasan mengenai landasan teori yang berkaitan dengan uji simulasi dan analisis yang akan dilakukan. Adapun hal – hal yang akan dijabarkan dalam bab ini yakni materi tentang jaringan MANET, *routing protocol* AODV, *Transmission Control Protocol, congestion control* (fase *slow start,* variabel *congestion window,* variabel *slow start threshold,* TCP *Tahoe, Reno* dan *New Reno*), QoS (*end to end delay, jitter, packet delivery ratio, throughput*), serta *Network Simulator-2* sebagai aplikasi penunjang untuk melakukan simulasi dan analisis jaringan. Materi tersebut dibutuhkan sebagai penunjang pembangunan simulasi dan analisis jaringan *Ad – Hoc* terhadap fase *slow start* sehingga tugas akhir ini dapat dibuat sesuai dengan rancangan yang diharapkan.

2.1. Jaringan Ad – Hoc

Jaringan *Ad – Hoc* adalah jaringan *wireless* yang terdiri dari kumpulan *mobile node* atau juga bisa disebut dengan *mobile station* yang bersifat dinamik dan spontan serta dapat diaplikasi di manapun tanpa menggunakan jaringan infrastruktur yang telah ada, seperti *access point,* seluler ataupun PSTN. Jaringan *ad – hoc* sendiri dibagi menjadi 7 jenis, [12] yakni :

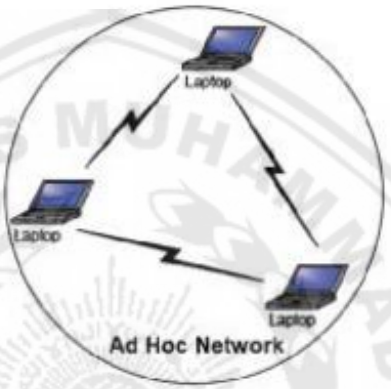
a. WANET (*Wireless Ad - Hoc Network*)
b. VANET (*Vehicular Ad - Hoc Network*)
c. MANET (*Mobile Ad - Hoc Network*)
d. SPANs (*Smart Phone Ad - Hoc Network*)
e. iMANETs (*Internet Based Mobile Ad - Hoc Network*)
f. Military / Tactical MANETs
g. SPAN (*Self Powered Ad - Hoc Network*).

Pada jaringan *Ad – Hoc* setiap *node* tidak hanya berperan sebagai *sender* dan *receiver* informasi, melainkan juga berfungsi sebagai pendukung jaringan seperti *router*. Maka dari itu diperlukan *routing protokol* yang ditanamkan pada jaringan *ad – hoc*. Dalam melakukan pemilihan jenis *routing* pada jaringan *ad – hoc* harus ditentukan berdasarkan permasalahan bagaimana cara menentukan dan memperhitungkan rute yang paling baik, dan sebagainya. Semisal menggunakan metode *distance vector routing* [3], metode ini mempunyai keunggulan dalam

memelihara jarak dengan arah dan tujuan data. Setiap *node* secara periodik melakukan pertukaran jarak vektor dengan *node* tetangganya. Saat *node* tersebut menerima jarak dari vektor tetangganya, maka *node* akan melakukan perhitungan rute yang baru dan melakukan pembaharuan jarak vektor yang baru.

2.1.1. Topologi Ad – Hoc

Topologi jaringan *ad – hoc* dapat terbentuk ketika antara *node* ataupun *device* yang memiliki *WiFi adapter* saling terhubung tanpa melalui perantara *access point,* sehingga aliran data yang dikirim langsung diterima oleh *device* tujuan tanpa melalui perantara.



Gambar 2. 1. Ilustrasi Jaringan Ad – Hoc [14]

Pada **Gambar 2.1** mengGambarkan bagaimana jaringan *Ad – Hoc* diterapkan pada perangkat *mobile*¸ dan semua perangkatnya saling terhubung secara nirkabel melalui *access point* yang digunakan satu sama lain.

Penerapan topologi *Ad – Hoc* sendiri memiliki beberapa kelebihan, yakni :

a. Hemat biaya, dikarenakan perangkat tambahan yang diperlukan untuk berkomunikasi dapat diminimalisir.
b. *Rate Throughput* antar adapter dua kali lebih cepat daripada menggunakan menggunakan perantara *access point* dalam infrastruktur topologinya.

Topologi *ad – hoc* juga memiliki kekurangan yang dapat mengganggu dan menyebabkan transfer data yang tidak optimal, diantaranya :

a. Kapasitas *bandwidth* yang sangat terbatas sehingga memperlambat proses transfer data.
b. Jangkauan sinyal yang dipancarkan oleh adapter *wireless* tiap *node* biasanya terbatas atau tidak terlalu jauh.

c. Kemanan sangat kurang karena masih menggunakan metode WEP, berbeda dengan infrastruktur yang menggunakan metode WPA atau WPA2.

2.2. Mobile Ad – Hoc Network

Mobile Ad – Hoc Network adalah sebuah jaringan yang terdiri dari kumpulan *node – node* dengan sifat *mobilitas* yang fleksibel, di mana *node – node* tersebut dapat berkomunikasi dengan tanpa menggunakan jalur komunikasi yang permanen, atau bersifat sementara (*Ad – Hoc*) seperti yang diilustrasikan pada **Gambar 2.2** [12].

Gambar 2. 2. Ilustrasi Jaringan Mobile Ad – Hoc Network [14]

2.2.1. Kelebihan Mobile Ad – Hoc Network

Dengan mengacu teori dari jaringan *ad – hoc* yang sebelumnya dijabarkan, *mobile ad – hoc* sendiri mempunyai beberapa kelebihan, yakni :

a. Cepat dalam beradaptasi terhadap perubahan topologi yang terjadi di jaringan.
b. Menerapkan konsep *Hand – Off Management.*
c. Memiliki tingkat fleksibilitas tinggi ketika menentukan topologi dan tidak teralu lama menyesuaikan dengan kondisi di lapangan atau kondisi yang sebenarnya.

2.2.2. Karakteristik Mobile Ad – Hoc Network

Mengacu pada ciri dasar utama pada MANET, MANET terdiri dari *mobile platform* atau bisa disebut dengan "*node"* yang dapat berpindah – pindah ke mana saja atau kapan saja dan hal ini membuat sifat MANET menjadi dinamis seperti yang diilustrasikan pada **Gambar 2.3**. Setiap *node* dilengkap dengan *transmitter* dan *receiver wireless* dilengkapi dengan antenna yang bersifat *omnidirectional*

(gelombang radio dipancarkan ke satu arah tertentu), *highly directional*, memungkinkan untuk dimodifikasi arah sinyalnya, atau kombinasi dari beberapa hal tersebut [14].

Gambar 2. 3. Topologi Dinamis MANET [12]

Selain itu, MANET juga memiliki karakteristik di berbagai sektor, antara lain :

a. Topologi yang dinamis

   Node pada MANET mempunyai sifat yang dinamis, dapat berpindah – pindah ke mana saja. Artinya topologi jaringan di dalamnya nantinya berbentuk loncatan antara *hop* ke *hop* yang dapat berubah secara tidak terpola dan terus menerus tanpa adanya ketetapan waktu yang terpola juga.

b. Otonomi

   Tiap *node* pada MANET berfungsi sebagai *end – user* sekaligus sebagai *router* yang dapat menghitung sendiri *router – path* yang selanjutnya akan dipilih menjadi rute jalur komunikasi.

c. *Bandwidth* terbatas

   Pada dasarnya *link* pada jaringan *wireless* cenderung memiliki kapasitas yang cukup rendah jika dibandingkan dengan jaringan *wired.* Kapasitas

   yang keluar untuk komunikasi pada *wireless* juga cenderung lebih rendah dari kapasitas maksismum transmisi. Dampak yang akan terjadi pada jaringan yang mempunyai kapasitas rendah seperti demikian adalah terjadinya *congestion* pada jalur komunikasi data.

d. Energi terbatas

Semua *node* pada topologi MANET memiliki karakteristik *mobile* menjadikan setiap *node* di dalamnya dipastikan menggunakan tenaga baterai dalam pengoperasiannya. Maka dari itu perlu adanya management optimalisasi energi.

e. Tingkat kemanan yang terbatas

Pada dasarnya, jaringan *wireless* cenderung lebih rentan terhadap kemanan daripada jaringan yang berkabel. Jenis pencurian seperti *eavesdropping, spoofing,* atau *denial of service* harus lebih diperhatikan.

2.2.3. Fokus Pengembangan MANET

Fokus penelitian yang menggunakan topologi *mobile ad – hoc network* mengacu kepada beberapa hal, antara lain :

a. *Security* dan *Reability*

Sistem keamanan pada jaringan *wireless* sangat diperlukan dalam mencegah seseorang untuk mengambil dan mengirimkan paket yang tidak diinginkan. Selain itu, ketangguhan jaringan *wireless* pada dasarnya memiliki jangkauan yang terbatas.

b. *Routing*

Topologi pada MANET yang dinamis dan dapat berubah – ubah menciptakan tantangan untuk mencari solusi untuk melakukan *routing* paket. *Routing* sangat penting diterapkan pada MANET ketika ada perubahan posisi *node* yang memungkinkan besar jalur *routing* juga perlu dirubah.

c. *Quality of Service*

Implementasi QoS pada jaringan MANET yang selalu berubah – ubah merupakan tantangan sendiri. Penerapan QoS perlu dikembangkan agar dapat menyesuaikan dengan kondisi jaringan pada MANET.

d. *Power Consumption*

Sebagian besar pada perangkat *mobile* saat ini menggunakan baterai sebagai sumber dayanya. Perlu dikembangkan cara memperpanjang waktu operasi perangkat, salah satunya dengan cara memperbesar kapasitas baterai atau memperkecil jumlah konsumsi pada baterai.

e. *Internetworking*

Selain digunakan untuk berkomunikasi antar *node* di dalam MANET, juga diperlukan pengembangan teknologi untuk berkomunikasi pada jaringan tetap [14].

2.2.4. Routing Protocol MANET

Proses *routing* pada MANET tidak sama dengan apa yang terjadi pada jaringan jaringan berkabel. Hal ini dikarenakan setiap *node* pada MANET juga memiliki peran sebagai *router* yang menerima aliran data dan menghitung jalur yang tepat untuk selanjutnya dikirimkan ke jalur tersebut sampai tujuan. Tiap – tiap *node* dipengaruhi oleh *node* lain (otonomi) dalam menentukan teknik *routing.*

Terdapat berbagai jenis *routing protocol* pada MANET, dan secara keseluruhan dapat dibagi menjadi beberapa kelompok, yakni :

2.2.4.1. Proactive Routing

Pada algoritma ini bekerja mengelola daftar tujuan dan rute terbaru masing – masing dengan cara mendistribusikan *routing table* ke seluruh jaringan, sehingga jalur lalu lintas (*traffic*) akan sering dilalui oleh *routing table* tersebut. Permasalahan yang sering terjadi pada algoritma ini adalah restrukturisasi *routing table* yang akan berdampak pada melambatnya aliran data [15]. Berikut adalah contoh *routing protocol* dari algoritma ini :

a. B.A.T.M.A.N (*Better Approach to Mobile Ad Hoc Network*)
b. DSDV (*Highly Dynamic Destination Sequance Distance Vektor Routing Protocol*)
c. HSR (*Hierarchi State Routiing Protocol*)
d. IARP (*Intrazone Routing Protocol*)
e. LCA (*Linked Cluster Architecture*)
f. WAR (*Witness Aided Routing*)
g. OLSR (*Optimized Link State Routing Protocol*)

2.2.4.2. Proactive Routing

Algoritma ini akan mencari rute (*on demand*) dengan cara membanjiri jaringan dengan paket *router request*, sehingga dapat menyebabkan jaringan akan penuh (*clogging*) [15]. Berikut adalah *routing protocol* dari algoritma ini :

a. SENCAST
b. *Reliable Ad Hoh On Demand Distance Vector Routing Protocol*

c. *Ant-Based Routing Algorithm for Mobile Ad Hoc Network*
d. *Admission Control Enabled On Demand Routing (ACOR)*
e. *Ariadne*
f. *Associativity Based Routing*
g. *Ad Hoc On Demand Distance Vector (AODV)*
h. *Ad Hoc On Demand Multipath Distance Vector*
i. *Backup Source Routing*
j. *Dynamic Source Routing (DSR)*
k. *Flow State in the Dynamic Source Routing*
l. *Dynamic MANET On Demand Routing (DYMO)*

2.2.4.3. Flow Oriented Routing

Algoritma ini mencari rute dengan mengikuti aliran yang disediakan, yaitu dengan cara melakukan *unicast* secara terus menerus ketika meneruskan data saat mempromosikan *link* baru [15]. Berikut adalah *routing protocol* dari algoritma ini:

a. *Interzone Routing Protocol (IERP)*
b. *Lightweight Underlay Network Ad Hoc Routing (LUNAR)*
c. *Signal Stability Routing (SSR)*

2.2.4.4. Hybrid

Algoritma ini dihasilkan dari menggabungkan antara *proactive routing* dengan *reactive routing* [15]. Berikut adalah *routing protocol* dari algoritma ini:

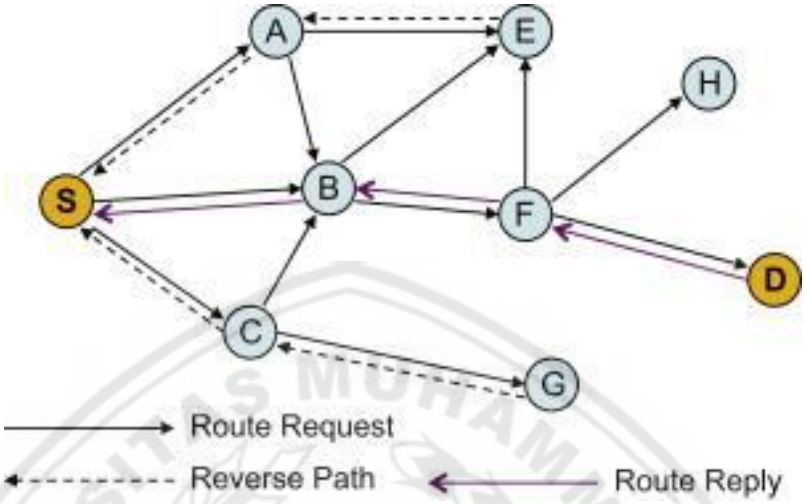
a. Hybrid Routing Protocol for Large Scale MANET (HRPLS)
b. Hybrid Wireless Mesh Protocol (HWMP)
c. Zone Routing Protocol (ZRP)

2.3. Protocol Ad Hoc on Demand Distance Vector (AODV)

Routing protocol AODV merupakan jenis algoritma yang bersifat *reactive.* Sesuai dengan karakteristik dari *protocol ractive,* algoritma ini mulai bekerja pada saat ada permintaan dari *source node* untuk mencari tahu jalur – jalur yang akan digunakan untuk mengirimkan pesan ke *node* tujuan. *Routing* ini akan menemukan jalur yang tidak ada atau tingkat *looping*-nya kecil dan menemukan jalur terpendek untuk menuju *node* tujuan [7][15].

Untuk menemukan jalur terbaik untuk *source node*, AODV akan melakukan *route discovery* dengan cara menyebarkan *route request* (RREQ) ke semua *node* yang bertetangga dengan *source node*. *Node* yang bertetangga tersebut akan

mengirimkan ke *node* tetangganya lagi sampai berakhir di *destination node.* Setelah status RREQ sampai di *node* tujuan, *node*tujuan akan membalas pesan RREQ dengan *route reply* (PREQ) sehingga terbentuklah jalur dengan rute terpendek dan memiliki *cost* yang lebih rendah daripada jalur yang lainnya seperti yang diilustrasikan pada **Gambar2.4**.



Gambar 2. 4. Route Discovery pada AODV [12]

2.4. Protokol Jaringan

Komputer satu dengan komputer lain dapat berkomunikasi satu sama lain ketika komputer tersebut terhubung dalam suatu jaringan yang mempunyai satu *set* peraturan yang sama [13]. *Set* peraturan yang dimaksud adalah protokol. Seperti analogi dua orang yang berlainan bangsa, maka untuk berkomunikasi diperlukan penerjemah (*nterpreter*) atau bahasa yang dimengerti oleh kedua belah pihak. Untuk menangani masalah komunikasi pada jaringan, maka dibentuklah model referensi OSI (*International Standardization Organization*) dan TCP/IP (*Trnasmission Control Protocol/Internet Protocol*).

2.4.1. OSI Reference Model

OSI merupakan model konseptual yang terdiri dari tujuh *layer* [13], masing – masing *layer* menetapkan fungsi jaringan tertentu. OSI mengGambarkan bagaimana informasi dari satu aplikasi komputer dibawa melintasi jaringan ke aplkasi yang sama di komputer lain.

OSI *reference model* membagi komunikasi jaringan menjadi tujuh *layer,* yakni :

a. *Application (layer 7)*
b. *Presentation (layer 6)*

c. *Session (layer 5)*
d. *Transport (layer 4)*
e. *Network (layer 3)*
f. *Data Link (layer 2)*
g. *Physical (layer 1)*

Semua lapisan dari model OSI dibagi lagi ke dalam dua kategori, yakni lapisan atas yang tediri dari *application, presentation, session,* dan *transport.* Lapisan atas berurusan dengan persoalan aplikasi dan pada umumnya diimplementasikan hanya pada perangkat lunak. Lapisan *application* merupakan lapisan penutup sebelum ke *user.* Proses *sser* dan aplikasi saling berinteraksi dengan perangkat lunak aplikasi yang berisi sebuah komponen komunikasi.

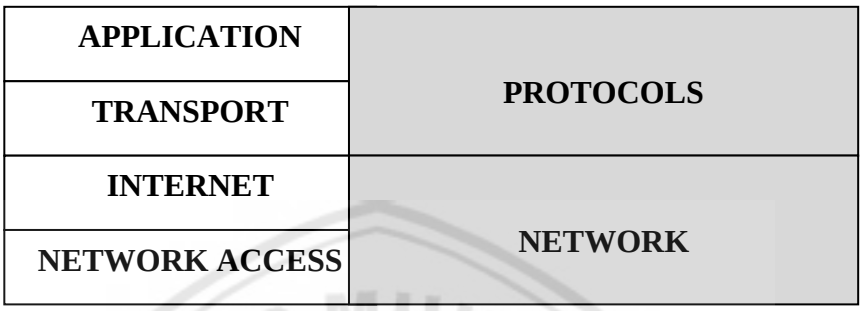
Lapisan bawah dari model OSI terdiri dari *network, data link,* dan *physical* yang mengendalikan persoalan pengiriman data. Lapisan bawah diimplementasikan ke dalam perangkat keras. Lapisan fisik adalah lapisan penutup pada media jaringan fisik, contohnya jaringan kabel, serta sebagai penanggung jawab bagi penempatan informasi pada media jaringan [13]. Berikut adalah penjelasan tujuh *layer* OSI :

Tabel 2.1. OSI *Layer*

Layer	**Keterangan**
Application	Membuka komunikasi dengan *user* lain serta mmeberikan layanan seperti *file transfer* maupun *e-mail* ke *user* lain dalam suatu jaringan.
Presentation	Berhubungan dengan perintah dari lapisan *application* dan melakukan *translation* antara tipe data yang berbeda jika diperlukan.
Session	Membuka, mengatur serta mematikan sesi antar aplikasi.
Transport	Menyediakan mekanisme untuk pembukaan, pengaturan serta penutupan jika ada permintaan dari sirkuit *virtual* pada data. Membuka end – to – end *connection,* serta menjaga pada data.
Network	Bertanggung jawab dalam *routing packet data* dari titik sumber ke titik tujuan serta bertanggung jawab dalam pengalamatan *node.*
Data Link	Menjaga sinkronisasi dan control kesalahan antar 2 pihak.
Physical	Menyediakan layanan transmisi *bit* melewati *channel* komunikasi secara elektrik, mekanisme, dan spesifikasi prosedur.

2.4.2. TCP/IP (Transmission Control Protocol/Internet Protocol)

TCP merupakan satu set standar aturan komunikasi data yang digunakan dalam proses transfer data dari satu komputer ke komputer lain di dalam jaringan komputer tanpa meilhat perbedaan jenis perangkat keras [13]. Model ini juga biasa disebut dengan *Internet Protocol Suite.* Model TCP/IP diilustrasikan seperti **Gambar 2.5** berikut :



Gambar 2. 5. Model TCP/IP [13]

Masing – masing dari lapisan TCP/IP mempunyai tugas komunikasi data sebagai berikut [13]:

a. *Network access layer*

   Lapisan ini memungkinkan menghubungkan *link* fisik ke media jaringan. Di dalam lapisan *network access* terdapat semua informasi detil mengenai *physical* dan *data link layer* pada model OSI termasuk detil teknologi WAN dan LAN.

b. *Internet layer*

   Lapisan ini bertujuan memilih jalur terbaik pada jaringan yang dapat dilewati oleh paket. Protokol utama yang dapat berfungsi pada lapisan ini adalah *Internet protocol.*

c. *Transport layer*

   Lapisan ini menyediakan koneksi logikal antara *host* sumber dan tujuan. Terdapat dua jenis protokol yang bekerja pada lapisan ini, yakni *Transport Control Protocol* dan *User Datagram Protocol.*

   - *Transmission Control Protocol* (TCP), protokol yang menyediakan layanan fitur *connection – oriented* dan *reliable,* yang menjamin data diterima *receiver* tanpa ada kesalahan, dengan urutan yang benar, dan tanpa ada duplikasi. Ada variabel yang mempengaruhi kerja TCP, yakni :

- *Acknowledgment* (ack) : apabila data sudah sampai pada suatu alamat yang tujuan, maka *source destination* akan memberitahu dengan mengirimkan ack ke *sender* bahwa data sudah diterima.
- *Sequence Number* : sistem penomoran yang diberikan kepada setiap paket data yang akan dikirimkan sehingga bisa diketahui data mana yang tidak sampai ke tujuan.
- *Windowing* : variabel ini mempengaruhi berapa besar paket data yang bisa dikirimkan dalam satu kali pengiriman paket sebelum menerima ack.

- *User Datagram Protocol* (UDP), sebuah protokol pada lapisan *transport* yang menyediakan layanan bersifat *connectionless* atau tidak *reliable.*

d. *Application layer*

Lapisan ini menangani representasi, *encoding* serta kontrol dialog. Model TCP/IP mengkombinasikan semua masalah yang berhubungan dengan aplikasi ke dalam satu lapisan ini.

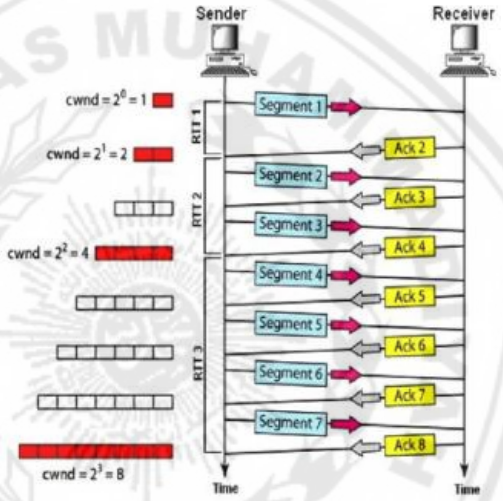
2.5. Congestion Control

Seperti pada penjelasan sebelumnya bahwa TCP menyediakan pengiriman paket yang bersifat *reliable*, sehingga aliran data yang dibaca TCP *receiver* tidak rusak, tanpa duplikasi, dan berurutan. Algoritma TCP *Congestion control* adalah faktor utama yang memegang peran penting dalam mempengaruhi performa dan sejumlah data yang dikirimkan dalam jaringan. *Congetion* terjadi ketika sejumlah paket melebihi kapasitas yang dapat diterima oleh *receiver* [1]. Namun TCP sendiri mempunyai mekanisme *congestion control* dalam menangani permasalahan tersebut. *Congestion control* memiliki mekanisme yang diklasifikasikan menjadi 4 fase utama:

a. *Slow Start*

Fase awal *congestion control* adalah fase *slow start*, di mana algoritma *slow start* digunakan oleh TCP *sender* menyesuaikan kecepatan aliran data ke *receiver* di mana periode *slow start* baru dimulai dengan mengamati setiap ACK yang diterima dari TCP *receiver*. Dengan demikian transmisi dari TCP *sender* sepenuhnya bergantung pada ACK yang dikembalikan oleh TCP *receiver*. Pada umunya setelah RTO (*Retransmission Time Out*), fase *slow start* diluncurkan di lokasi pengiriman koneksi TCP, dan setelahnya akan

secara signifikan mempengaruhi kinerja koneksi secara keseluruhan. Pada dasarnya, tujuan mekanisme *slow start* adalah membantu *sender* memperoleh *bandwidth* pada jalur jaringan. Pada **Gambar 2.6,** *slow start* dimulai dengan satu segmen untuk cwnd (cwnd = 1 segmen), dan setiap ack diterima secara berturut – turut, cwnd meningkat sebesar satu (cwnd baru = cwnd sebelumnya + 1). Pertumbuhan *congestion window* secara eksponensial dengan masing – masing *round trip time* akan menduplikasi ukuran *congestion window* (cwnd baru = cwnd sebelumnya x 2), sampai cwnd mencarapi titik *congestion* di *link.* TCP *slow start* diterapkan dengan dua variabel di dalamnya, *congestion window* (cwnd) dan *slow start threshold* (ssthresh).



Gambar 2. 6. Ilustrasi Kinerja TCP Slow Start [1]

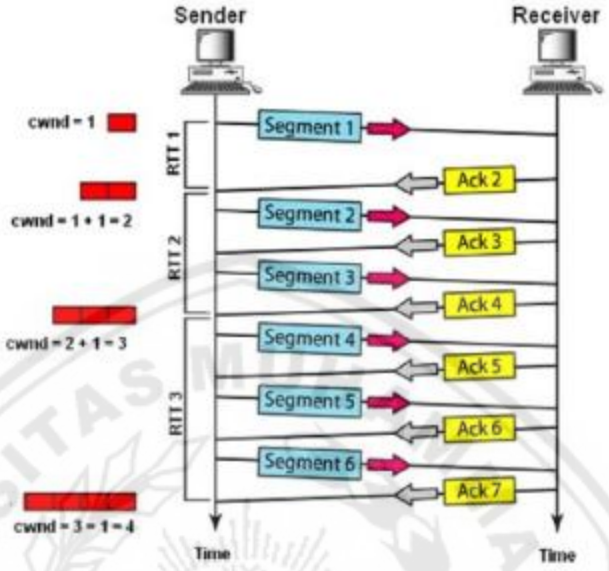
Variabel ssthresh merupakan nilai *threshold* di mana TCP meninggalkan periode *slow start.* Apabila pada suatu titik waktu ketika cwnd meningkat hingga di luar sshtresh, maka periode TCP pada titik waktu tersebut keluar dari fase *slow start.* Pada titik tersebut, TCP *source* masuk ke tahap berikutnya, yaitu fase *congestion avoidance* [1]. Ketika TCP keluar dari fase *slow start*, koneksi TCP sudah dalam keadaan stabil di mana *congestion window* secara hati – hati meningkatkan besar jalur jaringan.

b. *Congetion Avoidance*

Tiap nilai *threshold* melebihi ssthresh, TCP akan memperlambat laju peningkatan ukuran *window*. Mekanisme *congestion avoidance* diimplementasikan saat ukuran *congestion window* lebih besar daripada *threshold slow start.* Pada *congestion avoidance*, cwnd diperbesar satu

segmen setiap RTT dan kemudian mengurangi setengah dari sebelumnya ketika TCP *sender* merasakan *congestion* (*packet loss*) [1].

Pada **Gambar 2.7** mengilustrasikan bahwa TCP *sender* menggandakan ukuran *congestion* untuk tiap RTT hingga ukuran *window* lebih besar dari ssthresh. Dari titik itu, cwnd meningkat satu segmen setiap RTT.



Gambar 2. 7. Ilustrasi Congestion Avoidance [1]

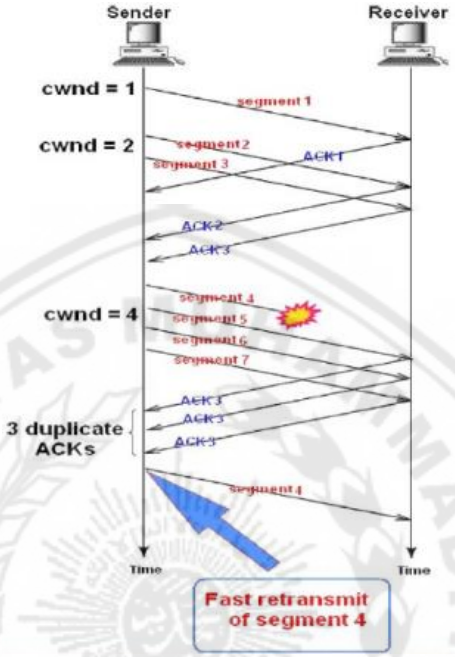
c. *Fast Recovery* dan *Fast Restransmit*

Sebenarnya kedua fase ini dapat digabung menjadi satu karena fungsi inti pada fase ini adalah untuk mempercepat *retrieval connection* atau pengambilan koneksi. Secara khusus, *fast recovery* digunakan oleh TCP untuk menghindari *waiting time* pengiriman ulang *time out* setiap segmen *loss* [1]. *Fast recovery* menyesuaikan pengiriman segmen baru sampai *non duplicate* ACK diterima oleh *receiver.* Bisa diilustrasikan setelah menerima paket (urutan nomor 1), *receiver* mengirimkan ACK dengan menambahkan nilai 1 ke urutan nomor (nomor urut = 1+1=2), berarti *receiver* menerima paket nomor 1 dan nomor 2 dari *sender*.

Dengan mengasumsikan tiga paket berikutnya sudah hilang, *receiver* menerima nomor paket 5. Setelah menerima paket nomor 5, *receiver* mengirimkan sebuah ACK dengan urutan nomor 2 dan 6. Ketika *receiver* menerima paket nomor 6, *receiver* akan mengirimkan ACK paket urutan nomor 2 dan 7. Dengan cara ini, *clocking ACK* dapat dijaga dan ketika *non duplicate* ACK sampai, *fast recovery* sudah selesai dan cwnd dikosongkan dan akan dilanjutkan dengan *fast retransmit.*

2.5.1. TCP Tahoe

TCP Tahoe dibuat berlandaskan pada algoritma TCP *congestion control.* TCP Tahoe menggunakan ack untuk mengeluarkan paket karena ack menandakan bahwa paket telah sampai pada penerima. TCP Tahoe memiliki 3 mekanisme, yakni *slow start, congestion avoidance,* dan *fast retransmit* [10].



Gambar 2. 8. Mekanisme fast retransmit [1]

Fase *slow start* pada TCP Tahoe dimulai ketika paket transmisi TCP diambil dari ACK yang masuk. TCP akan dimulai atau melakukan *restart* setelah *packet loss* harus melalui prosedur *slow start* dikarenakan pengiriman paket yang terlampau besar akan membanjiri jaringan sehingga koneksi mungkin tidak akan pernah dimulai. *Slow start* mengGambarkan bahwa pengirim mengatur cwnd menjadi 1, kemudian untuk tiap ACK yang diterimanya mengingkatkan cwnd dengan pertambahan 1 sehingga dalam RTT pertama TCP mengirimkan 1 paket, di pengiriman ke dua meningkatkan 2, di pengiriman ke tiga meningkatkan 4. Peningkatan terjadi secara eksponensial sampai mencapai titik ssthresh.

Pada tahap *congestion avoidance* dimulai ketika cwnd sudah melebihi ssthresh, yang nantinya cwnd akan ditingkatkan secara linier sampai bertemu dengan sebuah *packet loss.* Pada tahap *fast retransmit* dimulai untuk mengurangi *waiting time* oleh pengirim sebelum melakukan *retransmit packet drop.* TCP pengirim akan melakukan pencatatan waktu untuk mengetahui segmen yang hilang.

2.5.2. TCP Reno

TCP Reno tetap menggunakan prinsip dasar TCP Tahoe, yakni pada fase *slow start, congestion avoidance,* serta *fast retransmit.* Namun, pada TCP Reno menambahkan mekanisme tambahan, *fast recovery* [10]. Pada TCP Reno mengharuskan mendapatkan ACK setiap kali segmen diterima. Tiap kali menerima tiga duplikat ACK, maka dianggap sebagai tanda bahwa segmen hilang, jadi kembali mengirimkan segmen tanpa harus mengunggu *timeout.*

Pada fase *fast recovery,*mekanisme ini bertujuan untuk menjaga *throughput* tetap tinggi ketika terjadi *congestion.* Pada fase ini, ketika menerima tiga duplikasi ACK dan telah melakukan *fast retransmission,*maka TCP tidak masuk ke fase *slow start,* tetapu langsung masuk ke fase *congestion avoidance.*

2.5.3. TCP New Reno

Versi perobaan TCP *Reno* dikenal dengan nama TCP *New Reno* [2]. Algoritma *fast recovery* dari TCP New Reno adalah satu – satunya yang membedakan dengan TCP Reno. Ketika beberapa *packet loss* terjadi, TCP New Reno lebih dapat diandalkan daripada TCP Reno. Ketika beberapa paket duplikat diterima, TCP Reno memasuki *fast retransmit* dan tidak akan keluar dari fase *fast recovery.* Ketika terjadi beberapa *packet loss* di TCP New Reno, dibutuhkan sebuah indikasi bahwa paket – paket dalam urutan tersebut hilang dan paket – paket harus ditransmisikan ulang. Ketika beberapa paket hilang dari urutan data, TCP New Reno mengirim data tanpa *retransmission time out.* TCP New Reno adalah cara yang sangat efisien untuk menangani banyak *packet loss.* Ketika satu paket hilang dipertimbangkan, algoritma memiliki batasan untuk mendeteksi dan mengirim ulang paket per RTT. Jadi pada dasarnya, seperti dalam TCP Reno, TCP New Reno mengatasi pengurangan ukuran cwnd beberapa kali ketika terjadi *packet loss* [7]

2.6. Quality of Service (QoS)

QoS (*Quality of Service*) pada sebuah jaringan merupakan kemampuan dalam sebuah jaringan tersebut untuk menyediakan fitur dan layanan terhadap lalu lintas data jaringan tersebut [12]. Pada jaringan *ad – hoc,* QoS menjadi penting untuk diterapkan karena karakteristik dan kelebihan yang dimiliki jaringan *ad – hoc,* yakni *node – node* di dalamnya dapat bertambah dan berpindah secara fleksibel yang pada akhirnya berpengaruh terhadap jarak antar *node* dan *traffic* jaringan di dalamnya. Dengan demikian hal terssebut akan mempengaruhi kualitas layanan *network*

terhadap mobilisasi data di dalamnya. Dengan diterapkannya QoS pada jaringan *wireless ad – hoc* diharapkan layanan pada jaringan menjadi lebih baik dan terencana. Nilai variabel pada QoS meliputi *throughput, end to end delay, jitter,* serta *packet delivery ratio.*

2.6.1. Throughput

Throughput merupakan jumlah paket atau *bit* yang diterima dalam satuan waktu per detik melalui sebuah sistem atau media komunikasi [12]. Berbeda dengan *bandwidth, throughput* disa dikatakan sebagai kemampuan sebenarnya dalam suatu jaringan dalam melakukan pengiriman data secara berkala per satuan waktu. *Throughput* bisa diukur setelah transmisi dari suatu pengiriman paket / data, karena sistem akan menambah *delay* yang dapat disebabkan oleh *processor limitation, congestion, buffering inefficient, error transmission, traffic loads,* maupun bisa disebabkan dari desain perangkat keras yang tidak mencukupi dari suatu proses pengiriman paket data.

Poin utama terjadinya *throughput* adalah seberapa besar ketersediaan *bandwidth* yang cukup atau tidak untuk menjalankan suatu proses pengiriman paket data. Kalkulasi untuk menghitung perbandingan antara *bandwidth* dengan *throughput* adalah sebagai berikut [12] :

Rumus 2.1. Rumus perhitungan *throughput* dan *bandwidth* [12]

$$waktu\ download\ typical = \frac{ukuran\ file}{throughput} \quad (2.1)$$

$$waktu\ downlaod\ terbaik = \frac{ukuran\ file}{bandwidth} \quad (2.2)$$

2.6.2. End to End Delay

End to end delay merupakan waktu yang dibutuhkan paket dapat untuk mencapai tujuan, termasuk semua penundaan yang mungkin disebabkan oleh proses pencarian rute, serta antrian di proses awal. Perhitungannya dapat diketahui dengan kalkulasi mengurangkankan waktu pada saat paket pertama dikirimkan oleh sumber dan waktu di mana paket data pertama kali tiba di tujuan.

2.6.3. Jitter

Jiter adalah variasi dari *delay* atau selisih antara *delay* pertama dengan *delay* selanjutnya [12]. *Jitter* dapat terjadi karena beberapa faktor, diantaranya perangkat yang kurang mendukung, jumlah *host* dalam suatu jaringan, dan lain sebagainya.

Untuk mengatasi *jitter*, paket data yang dating dikumpulkan dalam *jitter buffer* selama waktu yang telah ditentukan sampai paket dapat diterima pada sisi *receiver*dengan urutan yang benar.

Rumus 2.2. Rumus perhitungan *jitter* [12]

$$jitter = \frac{total\ variasi\ delay}{(total\ paket\ data - 1)} \quad (2.3)$$

2.6.4. Packet Delivery Ratio

Packet delivery ratio merupakan perbandingan paket data yang diterima oleh tujuan untuk diolah yang dihasilkan oleh sumber paket data. Untuk menghitungnya dapat menggunakan kalkulasi sebagai berikut :

Rumus 2.3. Rumus perhitungan *packet delivery ratio* [12]

$$PDR = S1 / S2 \quad (2.4)$$

Dengan keterangan S1 adalah jumlah paket data yang diterima oleh masing – masing tujuan dan S2 adalah jumlah paket data yang dihasilkan oleh masing – masing sumber.

2.7. Buffer Management

Permasalahan yang sering terjadi pada aktivitas transmisi adalah adanya paket – paket yang besar datang pada jalur lalu lintas jaringan dan menyebabkan antrian yang besar dan kapasitas *buffer* akan mengalami kendala. Saat terjadi *packet flooding* yang terlalu besar maka *buffer* pada jalur jaringan akan penuh sehingga menimbulkan *packet loss.* Hal ini juga dipengaruhi oleh fase *slow start* dan algoritma manajemen antrian paket seperti *drop tail*.

Drop tail adalah mekanisme manajemen antrian yang bersifat pasif. Cara kerja manajemen ini yakni ketika *buffer* penuh, paket yang tidak tertampung dipastikan akan di *drop*. Ketika paket tersebut di *drop* maka koneksi TCP yang terhubung pada *link* akan dipaksa melakukan fase *slow start.*

2.8. Network Simulator – 2

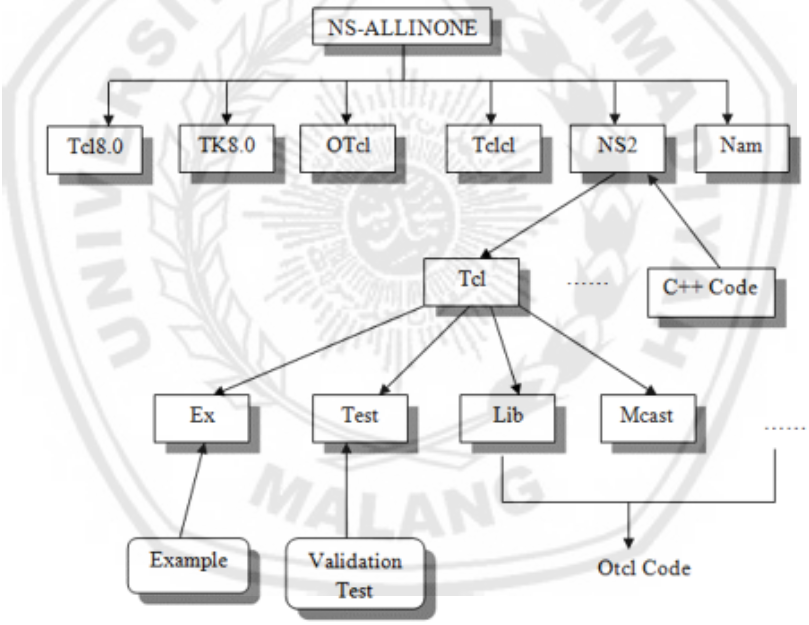
Network Simulator – 2 merupakan perangkat linak untuk kebutuhan simulasi palikasi, protokol, tipe jaringan pemodelan jaringan, elemen – elemen jaringan dan pemodelan lalu lintas jaringan. *Software* ini menggunakan dua buah bahasa pemrograman, yakni simulator berorientasi obyek yang ditulis dalam bahasa C++

dan OTcl (*Object Oriented Extension of* Tcl) untuk melakukan eksekusi terhadap peintah – perintah yang ditulis sebagai *script* dari NS-2 [12].

Penggunaan bahasa C++ pada *library* dikarenakan C++ mampu mendukung *runtime* simulasi yang cepat, meskipun simulasi melibatkan simulasi jumlah paket dan sumber data dalam jumlah besar. Namun demikian, bahasa Tcl memiliki respon yang lebih cepat dan interaktif ketka terjadi kesalahan *syntax* dan perubahan *script* meskipun memberikan respon *runtime* yang lebih lambat dari C++. Dengan fitur yang diberikan bahasa Tcl, *user* dapat mengetahui letak kesalahan yang dijelaskan pada *console* sehingga dapat langsung diperbaiki dengan cepat.

2.8.1. Komponen Network Simulator – 2

NS-allinone memiliki komponen yang wajib dan opsional yang dibutuhkan oleh simulasi. Masing – masing komponen di dalamnya terdapat pada folder *NS-allinone* yang di*install*.



Gambar 2. 9. Komponen Pembangun NS-2 [12]

Berikut adalah keterangan **Gambar 2.9** sebagai berikut :

a. TCL : *Tool Command Language*
b. Otcl : *Object TCL*
c. TK : *Toolkit*
d. Tclcl : TCL/C++ *Interface*
e. NS2 : versi simulator
f. Nam : *Network Animator* / *visualizer*